Self-imitation Learning for Action Generation in Text-based Games

Zijing Shi(^1), Yunqiu Xu(^1), Meng Fang(^2), Ling Chen(^1)

(^1) University of Technology Sydney, NSW, Australia
zijing.shi@student.uts.edu.au, {yunqiu.xu, ling.chen}@uts.edu.au
(^2) University of Liverpool, Liverpool, UK
Meng.Fang@liverpool.ac.uk

Abstract

In this work, we study reinforcement learning (RL) in solving text-based games. We address the challenge of combinatorial action space, by proposing a confidence-based self-imitation model to generate action candidates for the RL agent. Firstly, we leverage the self-imitation learning to rank and exploit past valuable trajectories to adapt a pre-trained language model (LM) towards a target game. Then, we devise a confidence-based strategy to measure the LM’s confidence with respect to a state, thus adaptively pruning the generated actions to yield a more compact set of action candidates. In multiple challenging games, our model demonstrates promising performance in comparison to the baselines.

1 Introduction

Text-based games are situated systems where the game agents observe textual descriptions, and generate textual commands to interact with the environment. These games have proven to be suitable test-beds for studying various natural language processing (NLP) tasks, such as question answering (Yuan et al., 2019; Xu et al., 2022), dialogue systems (Ammanabrolu et al., 2022), situated language learning (Shridhar et al., 2020) and commonsense reasoning (Murugesan et al., 2021; Ryu et al., 2022). Recent years have witnessed the thrivingness of designing reinforcement learning (RL) agents in solving these games (Narasimhan et al., 2015; Hausknecht et al., 2020; Ammanabrolu and Riedl, 2019; Xu et al., 2020b), while the combinatorial action space remains as a challenging issue, preventing RL agents from being deployed in real world applications.

In general, text-based games accept free-form actions, resulting in a large combinatorial action space. Fig. 1 shows a raw excerpt from the classic game “Zork1”. A 4-word action has to be selected from \(|V|^4\) candidates, where \(V\) denotes the vocabulary set (Xu et al., 2020a). Given that only 130 actions are required to solve this game, the agent wastes both training data and time in attempting irrelevant actions (Dulac-Arnold et al., 2015). To handle the combinatorial action space, early efforts either heavily rely on hand-crafted rules, or simply assume the availability of the action candidate set. For example, some works consider a set of currently admissible actions (He et al., 2016), or a template-based action space (Hausknecht et al., 2020). Alternatively, some other works alleviated this challenge by filtering inadmissible actions through methods such as action affordance (Jain et al., 2020), bandit-based elimination (Zahavy et al., 2018) and rule-based scoring (Ammanabrolu and Riedl, 2019).

---

Observation: The door reluctantly opens to reveal a rickety staircase descending into darkness.

Action: go down

Observation: The trap door crashes shut, and you hear someone barring it.

Action: go north

Observation: This is a small room with passages to the east and south and a forbidding hole leading west. A nasty-looking troll, brandishing a bloody axe, blocks all passages out of the room. Your sword has begun to glow very brightly.

Action: kill troll with sword

Observation: Almost as soon as the troll breathes his last breath, a cloud of sinister black fog envelopes him, and when the fog lifts, the carcass has disappeared. Your sword is no longer glowing.

---

Figure 1: Excerpt from the game “Zork1”. With a vocabulary size of 697, there are around \(697^4 \approx 200\) billion potential 4-word actions in the game.

---

Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, pages 703–726
May 2-6, 2023 ©2023 Association for Computational Linguistics
In handling the combinatorial action space for text-based games, recent pre-trained language models (PLMs) (Devlin et al., 2019; Radford et al., 2019; Brown et al., 2020; Andreas and Klein, 2016) can help generate actions. However, the potential of LM is still less effectively explored. As one of the pioneer works, Yao et al. (2020) proposed the CALM, which is a GPT-2 model pre-trained on human gameplay trajectories, to generate the action candidate set for the RL agent. However, when solving a previously unseen game, CALM tends to generate actions with less satisfying qualities, leading to two consequences that may affect RL training: 1) the action set may contain a large proportion of inadmissible actions, and 2) the useful actions may not be generated. As a mitigation, the CALM model is set to generate a relatively huge action candidate set, followed by ad-hoc operations to filter out the inadmissible actions, which requires prior knowledge. Micheli and Fleuret (2021) extended the LM-based agent to goal-conditioned tasks to follow instructions. Besides the offline pre-training data, the LM is further improved with the successful trajectories collected during online interaction. However, text-based games do not have well-defined goals. Furthermore, some games are so challenging that it is impossible to collect successful trajectories (Tuyls et al., 2022).

In this work, we address the crux of combinatorial action space in solving text-based games. We propose the Confidence-based Self-imitation Model (CSM) to generate the action candidates for the RL agent.* Firstly, we leverage the self-imitation learning method (Oh et al., 2018) to rank and exploit past trajectories of high values to adapt a pre-trained LM towards the target game. Then, we propose a confidence-based strategy to measure the LM’s confidence (Gandrabur et al., 2006) with respect to a state, thus adaptively pruning the action candidates based on the confidence value. Our model achieves promising performance in six challenging man-made games. Apart from significantly outperforming an action generation-based baseline, our strategy helps the RL agent to even achieve comparable performance to a baseline armed with the oracle action candidate set.

Our main contributions are summarized as follows: Firstly, we develop a LM-based framework to handle the issue of combinatorial action space in solving text-based games. Secondly, we propose a strategy to further improve the LM via self-imitation learning during the RL training. Thirdly, our experiments demonstrate that, the proposed method significantly improve the performance on multiple games compared with the strong contemporary method.

2 Related Work

2.1 RL Agents for Text-based Games

Inspired by the success of RL in playing games (Silver et al., 2016) and various NLP tasks (Fang et al., 2017; Yuan et al., 2019; Ammanabrolu et al., 2022), Narasimhan et al. (2015) and He et al. (2016) introduce RL to solve text-based games. Compared with non-learning-based agents (Hausknecht et al., 2019; Atkinson et al., 2019), the RL-based agents reduce the demand for extensive expert knowledge to develop gameplay strategies, and become the predominant modelling paradigm for solving text-based games. Subsequently, many variants of RL-based agents with different architectures and learning schemes have been proposed (Yuan et al., 2018; Jain et al., 2020; Guo et al., 2020; Xu et al., 2021; Tuyls et al., 2022; Shi et al., 2023). Innovations include modeling state space utilising knowledge graphs (Adhikari et al., 2020; Xu et al., 2020b), integrating question-answering and reading comprehension modules (Ammanabrolu et al., 2020; Xu et al., 2022). While these approaches focus on the problems of partial observability and language semantics, they still face the challenge of the combinatorial action space.

2.2 Combinatorial Action Space in TBGs

The combinatorial language-based action space is one primary challenge in solving text-based games. Early efforts mainly utilise hand-crafted rules or assume the agent has a predefined set of actions to choose from. For instance, the Jericho benchmark provides a valid action handicap that filters out inadmissible actions (i.e. actions that are either unrecognized by the game engine or do not change the underlying game state) at each game state (Hausknecht et al., 2020). This handicap has been widely used as the reduced action space by approaches like DRRN (He et al., 2016). In addition, the template-based action space is introduced where the agent selects first a template, and then a verb-object pair either individually (Hausknecht et al., 2020) or conditioned on the selected template (Ammanabrolu and Hausknecht, 2020). Even

*Code is available at https://github.com/winni18/CSM.
using the reduced action space, approaches filtering unnecessary actions can further improve the computational tractability and speed up the learning convergence (Zahavy et al., 2018; Jain et al., 2020).

2.3 Pre-training Methods for TBGs

Recent studies focus on enhancing the language understanding capability of agents by introducing pre-trained language processing modules. For instance, Singh et al. (2021) utilise the DistilBERT (Sanh et al., 2019) fine-tuned on human gameplay trajectories to represent game states. Ammanabrolu et al. (2020) employ the pre-trained ALBERT (Lan et al., 2019) to extract information from the textual observation by answering questions, and then update the knowledge graph during training. Adolphs and Hofmann (2020) use a pre-trained task-specific module to predict what is left to complete the tasks.

In general, RL-based agents are initialised with knowledge using pre-trained modules before exploring game environments.

Some studies leverage pre-trained language models for action generation (Hausknecht et al., 2020) or word embeddings for affordance detection (Fulda et al., 2017). The approach closest to our work is Yao et al. (2020), which is state-of-the-art without requiring access to admissible actions. In their study, a GPT-2 language model trained on human gameplay trajectories is used to generate action candidates for the RL agent to select. To ensure that the correct actions are provided, the GPT-2 model is set to generate a relatively huge action candidate set, followed by ad-hoc operations to predict the admissibility of an action based on environmental feedback. In contrast, our work intends to narrow down the action space via self-imitation learning and make learning tractable.

3 Preliminaries

Text-based Games as POMDPs The text-based game can be formally formulated as a partially observable Markov Decision Process (POMDP) \((S, T, A, O, R, \gamma)\). At each step \(t\), the agent receives a textual observation \(o_t \in O\) from the game environment, while the latent state \(s_t \in S\), which contains the complete internal information of the environment, could not be observed. By executing an action \(a_t \in A\), the environment will transit to the next state according to the latent transition function \(T\), and the agent will receive the reward signal \(r_t = R(s_t, a_t)\) and the next observation \(o_{t+1}\). The objective of the agent is to take actions to maximize the expected cumulative discounted rewards \(R_t = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t]\), where \(\gamma \in [0, 1]\) is the discount factor.

Trajectory and Episode We define the trajectory \(\tau\) as the sequence of observation-action pairs collected in an RL episode, i.e., \(\tau = (o_1, a_1, o_2, a_2, \ldots, o_l, a_l)\), where \(l\) is the length of \(\tau\). An RL episode is the process of an agent interacting with the environment from the beginning of a game to a termination state (e.g., the agent dies) or the step exceeding the pre-defined limit.

DRRN Existing RL methods for solving text-based games use game rewards to learn a value function. For instance, the Deep Reinforcement Relevance Network (DRRN) (He et al., 2016) is a choice-based game agent, where each action candidate \(a\) is paired with the state \(o\) to check its relevance. The agent then passes each pair through a deep neural network with parameters \(\phi\) to estimate the \(Q\)-values \(Q_\phi(o, a)\). The parameters \(\phi\) of DRRN are trained using tuples \((o, a, r, o')\) sampled from a prioritized experience replay buffer with the temporal difference (TD) loss:

\[
L_{TD}(\phi) = \left( r + \gamma \max_{a' \in A} Q_\phi(o', a') - Q_\phi(o, a) \right)^2
\]

where \(r\) is the game reward and \(\gamma\) is the discount factor. The next action is then selected by softmax sampling the predicted \(Q\)-values:

\[
\pi_\phi(a|o) = \frac{\exp(Q_\phi(o, a))}{\sum_{a' \in A} \exp(Q_\phi(o, a'))}
\]

To circumvent the challenge of combinatorial action space, DRRN assumes access to the valid action handicap provided by the environment at each game state.

4 Methodology

4.1 Overview

To address the combinatorial action space, we propose the Confidence-based Self-imitation Model (CSM), which leverages the advantages of pre-trained LM and Self-imitation Learning (Sil) for adaptive action generation. Fig.2 shows an overview of CSM. At time step \(t\), the LM is provided with the context \(c_t = (o_{t-1}, a_{t-1}, o_t)\) as the input, and generates a set of action candidates \(A_t\)
Figure 2: An overview of CSM. The LM takes the context $c_t$ and generates action candidates $A_t$, and conducts action pruning to further reduce the action space. The RL agent takes the observation $o_t$, and selects an action $a_t \in \hat{A}_t$. The valuable trajectories $\tau$ are collected to further improve the LM through self-imitation learning.

as well as their probabilities using beam search decoding. Based on the probabilities, we conduct Action Pruning (AP) to obtain a more compact subset of action candidates $\hat{A}_t \subseteq A_t$ for the RL agent. Then the RL agent considers the observation $o_t$ and selects an action $a_t \in \hat{A}_t$. To generate high-quality actions which are more context-relevant, we adapt the LM towards the target game during the RL training. Specifically, we collect and then select the past valuable trajectories $\tau$ in an additional replay buffer, to further improve the LM through a self-imitation learning manner.

4.2 Self-imitation Learning

We follow the work of Yao et al. (2020) to utilize the LM for action generation. During pre-training, given human gameplay trajectories $\tau$, we first build the context $c_t$, then train the LM to minimize the expected cross-entropy loss: $L_{LM} = -\mathbb{E}[\log p(a|c)]$, where $\log p(a|c) = \sum_{i=1}^m p(a_i|a^{<i},c)$ for an action with $m$ tokens. During RL, the LM will serve as a “rough” action selector to generate the top-$k$ actions. Then the RL agent will select one action to interact with the environment.

One drawback of the previous work Yao et al. (2020) is that when facing an unseen context, the LM may generate actions with poor performance. A straightforward solution is to continuously improve the LM during RL, thus making it adapted to the target game. Since no external trajectories (e.g., from human players) are available in the RL stage, we consider resorting to the self-imitation learning (Gangwani et al., 2019), i.e., letting the LM learn from the trajectories collected during the RL interaction. One thing we should pay attention to is the quality of the trajectories — sub-optimal trajectories may adversely affect imitation learning (Hu et al., 2019; Xu et al., 2022). Text-based games, especially games originally designed for human players, may be too challenging for agents to walk through. Thus, we cannot directly obtain successful trajectories during interacting with the environment. To alleviate this problem, we build a heap-like replay buffer to store past high-quality trajectories. We regard those obtaining higher scores with fewer steps as high-quality trajectories.

4.3 Confidence-based Action Pruning

Through the aforementioned SiL, the LM is expected to generate a more reliable action candidate set $\hat{A}_t$ of size $N$. For each action $a_{t,i} \in \hat{A}_t$, we then calculate its normalized probability $P(a_{t,i}|c_t)$.
Table 1: Game statistics.

<table>
<thead>
<tr>
<th>Game</th>
<th>Avg.Action Number</th>
<th>Avg.Action Length</th>
<th>Avg.Steps Per Reward</th>
<th>Walkthrough Length</th>
<th>Max Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Balances</td>
<td>23.29</td>
<td>2.99</td>
<td>12</td>
<td>122</td>
<td>51</td>
</tr>
<tr>
<td>Inhumane</td>
<td>6.96</td>
<td>2.36</td>
<td>14</td>
<td>123</td>
<td>90</td>
</tr>
<tr>
<td>Ludicorp</td>
<td>14.52</td>
<td>2.76</td>
<td>4</td>
<td>364</td>
<td>150</td>
</tr>
<tr>
<td>Snacktime</td>
<td>5.68</td>
<td>2.14</td>
<td>8</td>
<td>34</td>
<td>50</td>
</tr>
<tr>
<td>Zork1</td>
<td>15.96</td>
<td>2.75</td>
<td>9</td>
<td>400</td>
<td>350</td>
</tr>
<tr>
<td>Ztuu</td>
<td>33.93</td>
<td>2.96</td>
<td>5</td>
<td>84</td>
<td>100</td>
</tr>
</tbody>
</table>

according to the beam search score. The probabilities exhibit two characteristics: 1) the long-tail phenomenon in linguistics (Zhan et al., 2021), where only a few probabilities produce lots of actions; 2) the probability distribution varies greatly under different states. Given these findings, we adopt a confidence-based strategy to further prune action candidates of low values, aiming to obtain a further reduced action candidate set \( \hat{A}_t \subseteq A_t \). Specifically, we accumulate the probabilities of top-\( k \) action candidates as the confidence value: \( \text{Conf}_t(k) = \sum_{i=1}^{k} P(a_{t,i} | c_t) \). We then conduct action pruning (i.e., constraining the action space \( k \)) by bonding the confidence value to a fixed, manually determined threshold \( \xi: \hat{A}_t = \{a_{t,i} | \text{Conf}_t(k) \leq \xi \} \). In this way, top-\( k \) action candidates are selected adaptively. For a more “familiar” context \( c_t \) (e.g., it is similar to a context that LM has encountered before), the LM is supposed to be able to obtain correct actions from the training data, and the probability distribution will be centralised to top-ranked actions. In contrast, for an “unfamiliar” \( c_t \), the actions’ probabilities might be more uniformly-distributed. In this case, the size of action candidates (e.g., \( k \)) will be expanded to ensure a high confidence value.

5 Experiments

5.1 Experimental Setup

We conduct experiments upon six games provided by the Jericho Game Suite (Hausknecht et al., 2020). These games have diverse themes and genres, and each of them can represent a type of task. Different from those generated through pre-defined simple rules (Côté et al., 2018), the games we use are more complex, making them even challenging for human players. Some games contain nonstandard actions (e.g., the spells), which are unlikely to be understood by the language model pre-trained with commonsense knowledge. Table 1 shows the game statistics calculated from the walkthrough of each game.

5.2 Baselines

Our work focus on the challenge of combinatorial action space in text-based games. Thus, we compare CSM with two baselines:

- **CALM** (Yao et al., 2020), which is a pioneer work in LM-guided action generation.

- **DRRN** (He et al., 2016), which assumes access to the “oracle” action set (i.e., the valid action handicap provided by the environment).

Of these methods, CALM is the previous state-of-the-art model without the availability of “oracle” action sets, while the DRRN agent with “oracle” action sets can be regarded as our “upper bound”.

5.3 Implementation Details

**Training** We implement CSM upon CALM’s released code, including a pre-trained GPT-2 LM †. Both CSM and CALM adopt DRRN as the RL agent, except that \( \hat{A}_t \) is obtained by LM. We set the step limit of an RL episode as 100, and train the RL agent on 8 parallel running environments for 100k steps. For each step, we train the RL agent with a batch size of 64, using an Adam optimizer with a learning rate of 1e-4. We set the first 20k steps as the warm-up phase, and start self-imitation learning as well as action pruning after this phase.

For SiL, we use a trajectory buffer with a size of 50. For every 500 steps, we update the LM for 1 epoch with a batch size of 8, using an Adam optimizer with a learning rate of 2e-5. If there are no fresh trajectories as the training progresses, we conduct SiL using existing trajectories within the buffer. For AP, we use beam search decoding with a beam size of 40 to generate actions and choose

†https://github.com/princeton-nlp/calm-textgame
Figure 3: The performance of CSM compared to baselines (CALM and DRRN) throughout training. Shaded areas indicate one standard deviation. Our CSM outperforms CALM while getting much closer to DRRN. Besides, it successfully solves the game “Snacktime”.

Table 2: The performance of CSM compared to baselines (CALM and DRRN) after training. The result with † is from Hausknecht et al. (2020). In six environments, our method obtains significant improvement compared to the CALM model, with an average normalized game score of 31.4%.

<table>
<thead>
<tr>
<th>Game</th>
<th>Generated $A_t$</th>
<th>Oracle $A_t$</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CSM</td>
<td>CALM</td>
<td>DRRN</td>
</tr>
<tr>
<td>Balances</td>
<td>11.7</td>
<td>10.5</td>
<td>14.0</td>
</tr>
<tr>
<td>Inhumane</td>
<td>27.0</td>
<td>20.6</td>
<td>33.6</td>
</tr>
<tr>
<td>Ludicorp</td>
<td>9.8</td>
<td>6.8</td>
<td>17.5</td>
</tr>
<tr>
<td>Snacktime</td>
<td>49.8</td>
<td>24.0</td>
<td>20.0</td>
</tr>
<tr>
<td>Zork1</td>
<td>40.6</td>
<td>34.3</td>
<td>40.0</td>
</tr>
<tr>
<td>Ztuu</td>
<td>17.5</td>
<td>11.7</td>
<td>21.6†</td>
</tr>
<tr>
<td>Avg.Norm</td>
<td>31.4%</td>
<td>19.6%</td>
<td>24.9%</td>
</tr>
</tbody>
</table>

The top 30 actions, i.e., $N = 30$. Then, we use the proposed confidence-based strategy to keep top-$k$ highest-scoring action candidates ($k<30$). We set $\xi$ as 0.6, and bound $k$ to be no lower than 10. Following previous works, we define the score as the sum of rewards collected within an episode, and report the score averaged over the last 100 finished episodes.

**LM** For both CSM and CALM, we use the pre-trained GPT-2 model provided by Yao et al. (2020) as the LM module. The LM consists of 12 layers, 768 hidden sizes, and 12 attention heads. This module is first pre-trained on the WebText corpus (Radford et al., 2019), then re-trained on the ClubFloyd dataset (Yao et al., 2020), which consists of 426 human game playing transcripts on 590 games (note that the Jericho-supported games that we experiment with are not included).

**RL** Both CSM and CALM adopt the DRRN as the RL agent, except that the action candidate set is generated by the LM module. Given the current observation $o_t$, and a set of currently admissible actions $A_t$, the RL agent first encodes $o_t$ to build the state representation, then pairs it with each ac-
Figure 4: Average episode score throughout training for ablation models. Shaded areas indicate one standard deviation.

Table 3: Average episode score after training for ablation models. Overall, both the SiL and AP are crucial to our framework.

<table>
<thead>
<tr>
<th>Game</th>
<th>CSM</th>
<th>w.o. AP</th>
<th>w.o. SiL</th>
<th>constant AP</th>
<th>CALM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Balances</td>
<td>11.7</td>
<td>11.1</td>
<td>7.0</td>
<td>11.2</td>
<td>10.5</td>
</tr>
<tr>
<td>Inhumane</td>
<td>27.0</td>
<td>23.6</td>
<td>1.5</td>
<td>1.1</td>
<td>20.6</td>
</tr>
<tr>
<td>Ludicorp</td>
<td>9.8</td>
<td>6.6</td>
<td>8.6</td>
<td>10.0</td>
<td>6.8</td>
</tr>
<tr>
<td>Snacktime</td>
<td>49.8</td>
<td>37.1</td>
<td>10.5</td>
<td>18.8</td>
<td>24.0</td>
</tr>
<tr>
<td>Zork1</td>
<td>40.6</td>
<td>35.6</td>
<td>34.1</td>
<td>15.2</td>
<td>34.3</td>
</tr>
<tr>
<td>Ztuu</td>
<td>17.5</td>
<td>12.1</td>
<td>12.9</td>
<td>15.1</td>
<td>11.7</td>
</tr>
<tr>
<td>Avg.Norm</td>
<td>31.4%</td>
<td>24.8%</td>
<td>10.8%</td>
<td>14.5%</td>
<td>19.6%</td>
</tr>
</tbody>
</table>

5.4 Results

Fig. 3 shows the average episode score throughout training for the baselines, and Table 2 shows the average episode score after training for the baselines. Our CSM demonstrates its effectiveness by significantly outperforming the backbone CALM in all of the six games, with an average normalized game score of 31.4%. Given that DRRN has access to the “oracle” action set $A_t$, its performance can be regarded as our “upper bound”. We observe that the performance of CSM is much closer to DRRN, and even surpasses DRRN in two games. In particular, while DRRN gets stuck in the game “Snacktime”, CSM solves this game, making its average normalized score among the highest of all. In Sec. 5.6, we further discuss this case by analyzing
Figure 5: Sample gameplay from the game “Snacktime” along with the generated action candidates, and the action chosen by the RL agent (coloured with blue).

the underlying reasons.

5.5 Ablation Studies

In order to evaluate the contribution of the two components in CSM, we compare our model with two variants with either SiL (“w.o. AP”) or AP (“w.o. SiL”). In order to demonstrate the effectiveness of confidence-based AP, we also employ constant AP. We set $k$ to 12, which is the average number of actions selected by the confidence-based strategy. Fig. 4 shows the average episode score for the ablation models throughout training, and Table 3 shows the average episode score for the ablation models after training.

In general, adapting the LM with respect to the target game helps (“w.o. AP” v.s., “CALM”), while reducing the action space upon it further boosts the performance (“CSM” v.s., “w.o. AP”). Solely reducing the action space $\mathcal{A}_t$, in contrast, leads to poor performance (“CSM” v.s., “w.o. SiL” v.s., “CALM”). Also, simply utilizing the constant AP together with SiL results in a considerable performance drop. (“CSM” v.s., “constant AP”). Without SiL, the LM has a greater chance of incorrectly filtering actions that are essential to go through the target game.

5.6 Qualitative Analysis

To demonstrate the efficacy of the proposed framework, we present two gameplay examples from the game “Snacktime”. Fig. 5 shows the generated action candidates and the action chosen by the RL agent, where “Context” denotes $c_t$, “CSM” and “CALM” denote the actions generated by CSM and CALM respectively, “DRRN” denotes the “oracle” action set used by DRRN. In the first example, all models generate and select the correct action “jump on him”, which leads to a +10 reward. Compared with CALM, CSM successfully reduces the action set from 30 to 10, relieving the burden for the RL agent. In the second example, both CSM and CALM generate action sets with the correct action “chew wand” being included. We found that the “oracle” action set provided by the environment is not always perfect, which explains why DRRN gets stuck here.

Similar phenomenon has also been reported in some other games (Tuyls et al., 2022).

6 Conclusion

In this work, we studied reinforcement learning in solving the text-based game. We proposed the CSM framework to generate a set of action candidates for the RL agent, which alleviates the issue of combinatorial action space. During RL training, we...
collected and exploited past high-quality trajectories and utilised self-imitation learning to improve the language model. In addition, a confidence-based action pruning strategy was proposed to further restrict the action space. We evaluate our method using the Jericho benchmark. In a variety of text-based games, our method significantly improves the performance compared with the strong contemporary method, and even overcomes the challenging bottleneck in the game “Snacktime”.

Limitations

In terms of limitations, text-based games are still far from being solved. Even if the agent has access to admissible actions, sparse rewards, language semantics and partial observability remain challenging obstacles for the existing game agent. In this study, we develop an effective framework to solve the issue of combinatorial action space. Future work can integrate our framework with methods that better leverage linguistic signals in order to make further progress in solving text-based games.

Acknowledgements

We thank the anonymous EACL reviewers for their insightful comments and constructive feedback.

References


Xingdi (Eric) Yuan, Marc-Alexandre Côté, Alessandro Sordoni, Romain Laroche, Remi Tachet des Combes, Matthew Hausknecht, and Adam Trischler. 2018. Counting to explore and generalize in text-based games. In European Workshop on Reinforcement Learning (EWRL).


Appendix

The appendix is organized as follows: Sec. A shows more experiment results. Sec. B provides the interaction log of CSM on the game “Snacktime”.

A More Results

Reproduction of DRRN  Fig. 6 shows the reproducing result of the DRRN baseline, where “DRRN - Ours” denotes the “DRRN” used in the main paper. The dashed lines “DRRN - Official” denote the results reported in Hausknecht et al. (2020) and Yao et al. (2020). According to Tuyls et al. (2022), the action candidate set $A_t$ provided by the environment is not always perfect, so that they manually augmented the environment-provided $A_t$ with actions from the game walkthrough which are required for making progress§. We follow their setting to modify the environment and rerun the DRRN baseline, yielding much better performance than the official results except one game “Ztuu”, which we use the official result in Table 2.

Reproduction of CALM  Fig. 7 shows the reproducing result of the CALM baseline, where “CALM 20% WU - Ours” denotes the “CALM” used in the main paper. The dashed lines “CALM 100% WU - Official” denote the results reported in Yao et al. (2020). In terms of the original CALM, our replication results are comparable with or better than the official results (“CALM 100% WU” v.s., “CALM 100% WU - Official”). The original CALM adopts a fast-text model to filter out the inadmissible actions from $A_t$ throughout the RL training process (i.e., they conduct warm-up for 100k steps), heavily alleviating the problem of generating inadmissible actions (“CALM 100% WU” v.s., “CALM w.o. WU”). However, obtaining this fast-text model requires prior knowledge, such as the additional training data and annotations. In our work, we would like to reduce the requirement of such external knowledge, and let the LM to conduct action pruning by itself. For all LM-based models, we only conduct warm-up for the first 20k steps, and discard the fast-text model afterwards (“CALM 20% WU”). As a future direction, we would like to consider more advanced warm-up strategies (Zha et al., 2021), thus eliminating the need for pre-training the fast-text model.

More results  Besides the episode score, we provide more results for further analyzing self-imitation learning and action pruning. Regarding SiL, Fig. 8 and Fig. 9 show the average score and length of the trajectories collected in the ranked buffer, respectively. There’s no doubt that the average score grows higher as the agent makes progress. Diverse trends could be observed in terms of the average length, since a newly-added trajectory might have both high score and more steps. Regarding AP, Fig. 10 shows the number of LM generated actions $k$, i.e., $|\hat{A}_t|$, where it could be observed $k$ gets close to the lower bound after pruning. Fig. 11 shows the LM probability of the top-1 generated action, and Fig. 12 shows the LM probability sum of the top-5 generated actions. After self-imitation learning, the top actions account for a larger proportion of the probability, making it safer for filtering those with low probabilities.

§https://github.com/princeton-nlp/XTX
Figure 6: The reproducing result of the DRRN baseline, where “DRRN - Ours” denotes the “DRRN” used in the main paper. The dashed lines denote the results reported in Hausknecht et al. (2020) and Yao et al. (2020).

Figure 7: The reproducing result of the CALM baseline, where “CALM 20% WU - Ours” denotes the “CALM” used in the main paper. The dashed lines denote the results reported in Yao et al. (2020).
Figure 8: The average score of trajectories in the ranked buffer.

Figure 9: The average length of trajectories in the ranked buffer.
Figure 10: The number of LM generated actions $k$, i.e., $|\hat{A}_t|$.

Figure 11: The LM probability of the top-1 generated action.
Figure 12: The LM probability sum of the top-5 generated actions.
We provide the interaction log of our CSM on the game “Snacktime”, where “Context” denotes $c_t$, “Actions” denotes $\mathcal{A}_t$, “Action” denotes $a_t$, “Reward” denotes $r_t$ after executing $a_t$, “Score” denotes the episodic sum of the collected rewards, and “Done” indicates whether this episode is finished.

**Interaction Example**

Context56826: [CLS] reset [SEP] <RROWWGRROWL> Your stomach growls. You’ve been vegging out in front of that flickering screen for hours, your pet by your side. Now your complaining stomach sets you up on your feet, and on a mission. Snack Time!

An Interactive Break for a Bite by Hardy the Bulldog (with help from Renee Choba) Release 1 / Serial number 080929 / Inform 7 build 5U92 (16/v6.31 lib 6/12N) (For more information please type ABOUT)

Sitting Room This is the room where you sit a lot. Well, you sleep here sometimes too. But there is a different room that is just for sleeping, and it is to the north. There’s another room to the west. It’s the room with the food. That is a good room. Even though this is the sitting room you can’t sit on everything.

There

Actions56826: [“’s no [cls] [cls”, “’s [cls] [cls”, “’s no [cls] [cls”, “’s no [cls] [cls”, “are [cls] [cls’], “is no [cls] [cls’], “[cls] [cls’], “are no [cls] [cls’], “are [cls] [cls’]”

>> Action56826: are no [cls] [cls]

Reward56826: 0, Score 0, Done False

Context56827: [CLS] are no [cls] [cls [SEP] That’s not a trick you know. Right now you aren’t carrying anything. (It’s not like you need a lot of stuff, anyway.) [SEP]


>> Action56827: scratch man

Reward56827: 0, Score 0, Done False

Context56828: [CLS] That’s not a trick you know. [SEP] scratch man [SEP] You scratch your pet, just gently, not to hurt him or anything. Your pet twitches a little and stirs, then is still again. It’s a start at waking him up. Right now you aren’t carrying anything. (It’s not like you need a lot of stuff, anyway.) [SEP]


>> Action56828: push pet

Reward56828: 0, Score 0, Done False

Context56829: [CLS] push pet [SEP] You tuck your head up under your pet and push back and forth to shake him. Your pet gets one eye open and half heartedly pets you. You’re making progress. Right now you aren’t carrying anything. (It’s not like you need a lot of stuff, anyway.) [SEP]

You scratch your pet, just gently, not to hurt him or anything. Hmm, your pet seems to have turned that into part of his dream or something, because he sure didn’t move this time when you scratched him. Right now you aren’t carrying anything. (It’s not like you need a lot of stuff, anyway.)

You jump up and get your paws on your pet for a second before sliding back down. Your pet shifts and stretches, blinking. He moves around until he is mostly sitting up. "Hey Hardy," he mumbles. It might take a little more to get him up and moving, but at least now his eyes are open!

Your score has just gone up by ten points. Right now you aren’t carrying anything. (It’s not like you need a lot of stuff, anyway.)

You tug and pull at the little soft things until finally one breaks free from the bunch. As you dislodge it, something clatters to the floor. Ah ha! You’ve found the magic wand! Those little soft things were trying to eat it! "So that’s where that got away to," says your pet, snatching up the magic wand. He points it at the box of light and makes it even louder. Held carefully between your teeth is a square soft thing.

You walk to the food room, it’s to the west, and the sleeping room is to the north. Right now you aren’t carrying anything. (It’s not like you need a lot of stuff, anyway.)
Context56834: [CLS] west west west west west west west west west west Food Room Food room, glorious food room! Your bowl for food and water is here, but that’s only the beginning. There’s also a lot of secret places where your pet keeps food and things that go with food and things that make food better. There’s a thing that gets hot and a box that stays cold, and that is your favorite because you know there is always food inside there. A cloth hangs down from the door of the cold box. Held carefully between your teeth is a square soft thing. [SEP] Actions56834: ['open cold box', 'open box', 'open door', 'get cloth', 'open fridge', 'take cloth', 'pull cloth', 'open refrigerator', 'use wand on box', 'push cloth']

>> Action56834: open refrigerator
Reward56834: 0, Score 10, Done False

Context56835: [CLS] open refrigerator [SEP] This is a good trick! (As you open your mouth, you drop the square soft thing.) You pull on the cloth with your teeth and back up as the door opens. Inside you can see various leftovers, some jars, some sandwich fixings and one shiny can. Right now you aren’t carrying anything. (It’s not like you need a lot of stuff, anyway.) [SEP] Actions56835: ['get can', 'take can', 'get all from fridge', 'take all from fridge', 'take all', 'get all', 'get all from refrigerator', 'take all from refrigerator', 'get everything', 'take everything']

>> Action56835: take all from fridge
Reward56835: 0, Score 10, Done False

Context56836: [CLS] take all from fridge [SEP] leftovers: All the food in the cold box is up too high for you or closed away. Looks like you will need your pet to help you get a snack. Jars: Jars are no good for you, they just tease you by letting you see what’s inside. But you can’t get them open without your pet’s help anyway. Sandwich fixings: All the food in the cold box is up too high for you or closed away. Looks like you will need your pet to help you get a snack. Shiny can: You snatch up the shiny can with your mouth and hold on tight. Held carefully between your teeth is one shiny can. [SEP] Actions56836: ['east east', 'east', 'east east east', 'east east east east', 'east east east east east', 'east east east east east east', 'east east east east east east east', 'east east east east east east east east', 'east east east east east east east east east east']

>> Action56836: east east
Reward56836: 0, Score 10, Done False

Context56837: [CLS] east east [SEP] Before you leave the food room, you push the cold box door with your nose to close it. (It’s
important to conserve energy.) Sitting Room Your pet is here, staring at the box of light, holding on tight to the magic wand. Held carefully between your teeth is one shiny can.

Actions: ['feed pet', 'feed man', 'give can to pet', 'give can to man', 'wait', 'push wand', 'east', 'feed feed man', 'take can', 'down']

>> Action56837: give can to man
Reward56837: 10, Score 20, Done False

Context56838: [CLS] give can to man [SEP] Your pet smiles as he reaches for the can. "Good boy, Hardy," he says, scratching you behind the ears. Your tail starts wagging; you just can't help it. (You've always been a fool for a good behind the ears scratch.) The can hisses with releasing air as the top pops, and then your pets takes a few good gulps. He must have been really thirsty, because soon he stands up and walks off to the other room. From the other room, you hear your pet banging around and yelling about the dino. Your pet must have almost tripped on it. He's always almost tripping on little stuff like that. After a second, you hear a click as the door to the water room closes. Now you have the long soft thing all to yourself! [Your score has just gone up by ten points.] Right now you aren't carrying anything. (It's not like you need

Actions56838: ['to move', 'all', 'pet', 'your pet', 'much', 'a pet', 'anything', 'you', 'it', 'any', 'anymore']

>> Action56838: pet
Reward56838: 0, Score 20, Done False

Context56839: [CLS] pet [SEP] That's not a trick you know. Right now you aren't carrying anything. (It's not like you need a lot of stuff, anyway.) [SEP]

Actions56839: ['get all', 'take all', 'take everything ', 'get everything', 'take pet', 'get pet', 'take all from refrigerator', 'take all from fridge', 'get can', 'take can ']

>> Action56839: get everything
Reward56839: 0, Score 20, Done False

Context56840: [CLS] That's not a trick you know. [SEP] get everything [SEP] magic wand: You snatch up the magic wand with your mouth and hold on tight. Held carefully between your teeth is a magic wand. [SEP]

Actions56840: ['use wand on box', 'push wand', 'bite wand', 'chew wand', 'take wand', 'get wand', 'read wand', 'pull wand', 'lick wand', 'east ']

>> Action56840: chew wand
Reward56840: 0, Score 20, Done False

Context56841: [CLS] chew wand [SEP] Making sure you are facing the box of light, you carefully press down on the wand with your teeth and are delighted when the box of light flickers and changes! Now you see a bunch of tiny people running around chasing a little
bitty ball. Uh oh, someone got pushed. Ugh, the faces got all huge, and they don't look happy. held carefully between your teeth is a magic wand. [SEP]

Actions56841: ['use wand on box', 'bite wand', 'push wand', 'chew wand', 'scratch wand', 'take wand', 'get wand', 'read wand', 'pull wand', 'lick wand']

>> Action56841: push wand
Reward56841: 0, Score 20, Done False

Context56842: [CLS] push wand [SEP] Once again you chomp down to change what's inside the box. This time there are these guys poking at each other, kinda fighting and joking around and all. This is the sort of thing your pet seems to enjoy watching sometimes, people getting hit with food and slipping and falling down and stuff. Everybody is laughing and nobody ever seems to really get hurt, so you guess it’s OK. held carefully between your teeth is a magic wand. [SEP]

Actions56842: ['use wand on box', 'bite wand', 'push wand', 'chew wand', 'jump on man', 'take wand', 'get wand', 'read wand', 'pull wand', 'lick wand']

>> Action56842: chew wand
Reward56842: 10, Score 30, Done False

Context56843: [CLS] chew wand [SEP] You’ve got the hang of this trick now for sure! One more bite on the wand brings up something new. This is more like it! Inside the box now is this lady putting together some tasty looking food! You're almost drooling just watching. This has got to make your pet think about eating! [Your score has just gone up by ten points.] held carefully between your teeth is a magic wand. you finally hear a familiar rush of water from the little water room and, after a second, the sound of the door opening. [SEP]

Actions56843: ['use wand on box', 'south', 'southeast', 'northwest', 'southwest', 'out', 'northeast', 'west', 'take wand', 'east']

>> Action56843: northwest
Reward56843: 0, Score 30, Done False

Context56844: [CLS] northwest [SEP] If you want to go to the food room, it’s to the west, and the sleeping room is to the north. You finally hear a familiar rush of water from the little water room and, after a second, the sound of the door opening. held carefully between your teeth is a magic wand. from the other room, you hear your pet banging around and yelling about the dino. your pet must have almost tripped on it, he's always almost tripping on little stuff like that. your pet walks in, he starts to sit down on the long soft thing but stops when he sees what's in the box of light. that lady cooking is doing her job! you pet stands and watches, transfixed. he must be thinking of food! [SEP]

Actions56844: ['out', 'southeast', 'down', 'south', 'northwest', 'southwest', 'northeast', 'wait', 'east', 'take wand']

>> Action56844: northeast
If you want to go to the food room, it's to the west, and the sleeping room is to the north. From the other room, you hear your pet banging around and yelling about the dino. Your pet must have almost tripped on it. He's always almost tripping on little stuff like that. Your pet walks in. He starts to sit down on the long soft thing but stops when he sees what's in the box of light. That lady cooking is doing her job! You pet stands and watches, transfixed. He must be thinking of food! held carefully between your teeth is a magic wand. Your pet makes his way to the food room. Your pet starts rattling around in the food room. Yes! He must be making a snack! held carefully between your teeth is a magic wand. From the other room you can hear the happy sound of the cold box being opened.

If you want to go to the food room, it's to the west, and the sleeping room is to the north. Your pet makes his way to the food room. Your pet starts rattling around in the food room. Yes! He must be making a snack! held carefully between your teeth is a magic wand. From the other room you can hear the happy sound of the cold box being opened.

A magical, bumpy thing your pet uses to control the box of light. From the other room you can hear the sound of secret places opening and closing. held carefully between your teeth is a magic wand. You don't hear anything from the food room. But then, making a sandwich is not a very noisy job.

> Action: east
Reward: 0, Score 30, Done False
If you want to go to the food room, it's to the west, and the sleeping room is to the north. You don't hear anything from the food room. But then, making a sandwich is not a very noisy job. Held carefully between your teeth is a magic wand. You don't hear anything from the food room. But then, making a sandwich is not a very noisy job. [SEP]

Actions:
'south', 'down', 'take wand', 'southeast', 'northwest', 'out', 'northeast', 'southwest', 'read wand', 'east'

>> Action: down

奖励: 0, 分数: 30, 完成: False

If you want to go to the food room, it's to the west, and the sleeping room is to the north. From the other room, you hear the sound of the cold box closing. You're sure that in just a minute your pet will be leaving the food room with the sandwich! [SEP]

Actions:
'northwest', 'southeast', 'south', 'down', 'northeast', 'southwest', 'out', 'read wand', 'east', 'take wand'

>> Action: northwest

奖励: 0, 分数: 30, 完成: False

Food Room Your pet is here, working on making a snack. You can also see a square soft thing here. Your pet seems to have finished cleaning up after making the snack. He turns to leave. Now is the moment of truth. Held carefully between your teeth is a magic wand. Sandwhich in hand, your pet heads for the sitting room. You follow your pet as he makes his way to the long soft thing and starts munching away on the sandwich. In a moment, it is gone, along with your dream of the perfect snack. Looks like you'll have to wait until dinner to eat. Poor hungry puppy! You have missed your chance to snack. In that game you scored 30 out of a possible 50, in 25 turns. Would you like to restart, restore a saved game or quit? [SEP]
Actions56852: ['give sandwich to pet', 'give sandwich sandwich to pet', 'give sandwich sandwich sandwich to pet', 'give sandwich sandwich sandwich sandwich sandwich sandwich sandwich to pet', 'give sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich to pet', 'give sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich to pet', 'give sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich to pet', 'give sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich to pet']

>> Action56852: give sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich sandwich to pet

Reward56852: 20, Score 50, Done True